Reinforcement Learning

Reinforcement Learning is a standard paradigm in machine learning to solve *control problems*.

- It models agent learning in an *environment*.
- It allows learning by *trial and error*.
- It leads an agent to discover *optimal policies*.

The underlying idea is that by interacting with an environment, an agent models its behaviour with respect to the *structure* of the environment.

Strengths of Reinforcement Learning

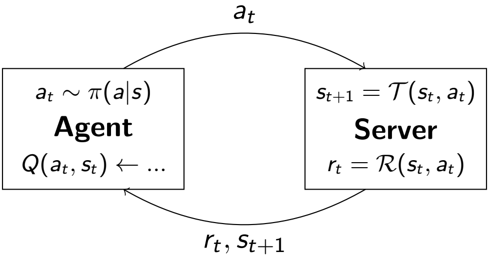
Reinforcement learning has been shown effective on several problems:

- Well-defined and well-structured synthetic environments (e.g.: boardgames)
- Non-stationary and adversarial synthetic environments (e.g.: real-time strategy games).
- Uncertain and varying real environments (e.g.: robot control).

Can we model hacking as a game learnable via *reinforcement learning*?

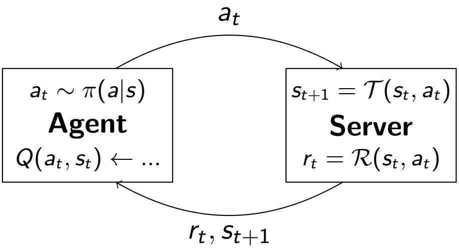
Modelling Hacking

We model *hacking* as a generic *game* (*capture-the-flag*), where the agent interact with a server trying to get a *flag*.



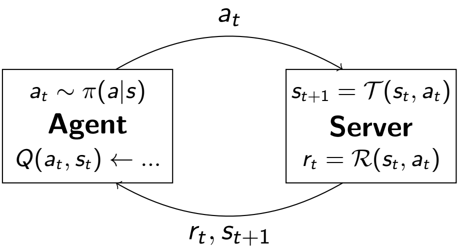
Hacking, though, raises some *unique* and *interesting* challenges when compared to classic RL applications.

Challenges (1)



- Security systems have *high entropy*: a real-world server tries to release as little information as possible to the agent.
- *The greatest challenge is NOT learning an optimal strategy, but discovering the structure of the server.*
- *Can we efficiently learn policies that are better than random guessing?*

Challenges (2)



- Human reasoning in hacking is *much more simple inference on actions and consequences*: a real-world agent relies on a vast repertoire of background knowledge, previous experience, intuition, original analogical reasoning, knowledge of human behavior.
- A reinforcement learning agent relies on a single channel: *rewards* observed in game.
- *What prior knowledge can we inject in a RL agent? How can we do it?*

Simulations

So far we have simple simulations:

- *Port Scanning*
- *Web Hacking*
- *Website Hacking*

Running these simulation with standard RL algorithms (*Q-learning* algorithms) allowed us to appreciate the challenge and the limits of learning in a *hacking* environment.

Simulations

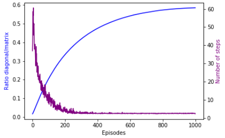


Figure: Reward and number of steps in a simple port scanning problem.

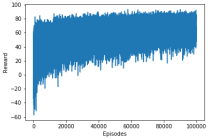


Figure: Reward in a more challenging web hacking problem.

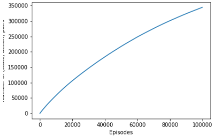


Figure: Size of Q-table in the challenging web hacking problem.

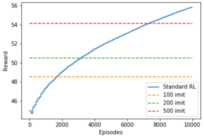


Figure: Benefit of imitation learning in a web hacking problem.

Future Challenges

We would like to develop our work from here:

- *Can we develop more realistic simulations?*
- *Can we solve real hacking problems?*
- *How can we adapt standard RL algorithms to the specific challenges of hacking?*